Quantum Genetic Optimization

Andrea Malossini, Enrico Blanzieri, and Tommaso Calarco

***Abstract*—The complexity of the selection procedure of a genetic algorithm that requires reordering, if we restrict the class of the possible fitness functions to varying fitness functions, is $\mathcal{O}(N \log N)$, where N is the size of the population. The quantum genetic optimization algorithm (QGOA) exploits the power of quantum computation in order to speed up genetic procedures. In QGOA, the classical fitness evaluation and selection procedures are replaced by a single quantum procedure. While the quantum and classical genetic algorithms use the same number of generations, the QGOA requires fewer operations to identify the high-fitness subpopulation at each generation. We show that the complexity of our QGOA is $\mathcal{O}(1)$ in terms of number of oracle calls in the selection procedure. Such theoretical results are confirmed by the simulations of the algorithm.**

***Index Terms*—Evolutionary computing and genetic algorithms, quantum computation.**

I. Introduction

QUANTUM algorithms exploit the laws of quantum mechanics in order to perform efficient computation. Such efficiency is granted when the algorithm is run on a quantum computer, whereas the simulation on a classical computer can be very resource-consuming. It has been shown that quantum computation can dramatically improve performance for solving problems like factoring [1] or searching in an unstructured database [2]. On the other hand, genetic algorithms [3] can be described, basically, as search algorithms. They work on a set of elements, called *population*, that evolves by means of crossover and mutation, towards a maximum of the fitness function. Since their proposition, genetic algorithms have proved to be efficient and flexible algorithms for solving a wide range of problems. Some attempts have been made in order to have fast hardware implementation of genetic algorithms [4]. In this perspective, having a quantum version of a genetic algorithm seems to be a relevant topic in the future, when quantum computers will be available. Moreover, the integration between the two paradigms can be a way of applying quantum computation to hard problems [5] for which a quantum algorithm is not available yet.

Manuscript received April 26, 2005; revised September 19, 2006.

A. Malossini is with the Department of Information and Communication Technology, University of Trento, I-38050 Povo, Trento, Italy (e-mail: malossin@dit.unitn.it).

E. Blanzieri is with the Department of Information and Communication Technology, University of Trento, I-38050 Povo, Trento, Italy (e-mail: blanzier@dit.unitn.it).

T. Calarco is with the Instituto Nazionale per la Fisica della Materia, Consiglio Nazionale delle Ricerche, BEC-INFM, I-38050 Povo, Trento, Italy, and with the European Centre for Theoretical Studies in Nuclear Physics and Related Areas, I-38050 Villazzano, Trento, Italy, and also with Institute for Theoretical Atomic, Molecular and Optical Physics, Harvard University, Cambridge, MA 02138 USA.

Digital Object Identifier 10.1109/TEVC.2007.905006

The possible interplay between quantum and genetic algorithms has been only partially explored. One of the first attempts to analyze benefits and drawbacks of a quantum approach to genetic algorithms is presented by Rylander *et al.* [6], where the elements of the population are quantum individuals (qubits). The qubit representation for the elements of the population is a key point for the use of the quantum algorithm. For example, by adopting a qubit chromosome representation, a classical population can be generated by repeatedly measuring the quantum population, and then its best elements are used to update the quantum population [7]. Other interesting approaches are to consider the elements of the population as quantum circuits and then to evolve them toward a target quantum circuit [8] or to use a quantum neural network to measure simultaneously the fitness values of all the possible elements of the population [9]. A recent survey on quantum genetic algorithms, in general, discussed some of the drawbacks of existing quantum genetic algorithms and presented some genetic algorithms for quantum circuit design [10]. Applications of quantum computation are wide-spreading in many different areas, for example quantum genetic algorithms for feature selection [11] or quantum algorithms for handling probabilistic, interval, and fuzzy uncertainty [12].

A promising area in which the combination of quantum computation and genetic algorithms can give advantages is that of applications with varying fitness functions. In these applications, the fitness function varies between genetic steps depending on some external time-dependent physical input. A very relevant example is given by noise in quantum control processes. In this scenario (already employed, in its classical version, in quantum chemistry experiments), genetic algorithms are used to select optimally shaped fields to drive a desired physical process, for instance, a laser-assisted molecular reaction [13], [14]. In such a case, the oracle consists of the physical process itself, rather than of a mathematical construction.

In this paper, we present a quantum genetic optimization algorithm (QGOA), a quantum algorithm that exploits the power of quantum computation in the fitness evaluation and selection procedures, and we show how to take advantage of quantum phenomena to efficiently speed up classical computation. In particular, we will see that the QGOA outperforms a classical genetic algorithm when the fitness function is varying [15] between genetic steps.

We exploit the power of quantum computation not only to represent the population by means of qubits, but also to perform fitness evaluation and selection. The algorithm is based on the Dürr–Høyer quantum algorithm for finding the minimum in an unsorted table [16]. Our results rely on the observation that it is possible to stop the quantum procedure of the Dürr–Høyer algorithm and to use the partial result for the selection. QGOA

TABLE I
NOTATION USED IN THIS PAPER

QGOA	Quantum Genetic Optimization Algorithm
$\mathbb{C}$	Complex space
$\|\cdot\rangle$	A Hilbert space vector
U	A unitary operator
U_F	Unitary operator for fitness evaluation
$\otimes$	Tensor product of Hilbert spaces
N	Number of elements of a population
M	Number of genetic steps
n	Number of qubits
$H^{\otimes n}$	n–qubit Hadamard–Walsh gate
n_{h}	Number of Dürr–Høyer iterations
κ_{BBHT}	Constant appearing in the BBHT algorithm
$f(\cdot)$	Fitness function
$F_i = F(i) = f(x_1)$	Fitness function computed on the element x_i
t	Number of marked solutions
$\theta(\cdot)$	Heaviside function
$\mathbb{E}(\cdot)$	Expectation value
$\sum_j$	$\sum_{j=0}^{N-1}$

uses the whole population at each genetic step, and in this sense it can be considered a "global search" algorithm.

A theoretical description of QGOA is provided, as well as a detailed analysis of the algorithm complexity. In particular, we show that the complexity of the quantum selection procedure (which includes the quantum fitness evaluation) does not depend on the size of the population N. Moreover, we show that the convergence speed, in terms of genetic steps, of the QGOA is comparable to the convergence speed of a classical steady-state genetic algorithm with truncation selection. Finally, we provide a simulation of the algorithm, which fully validates the theoretical results.

The remainder of the present section is devoted to introducing the concepts related to genetic and quantum computation that are necessary for presenting the algorithm. In Table I, we present the notations used in this paper. Section II presents our QGOA. Section III presents the analysis of the complexity, whereas Section IV is devoted to simulating the algorithm and to empirically validating the theoretical results. Finally, we draw some conclusions in Section V.

A. Introduction To Genetic Algorithms

Genetic algorithms are adaptive search algorithms based on the evolutionary ideas of natural selection and genetics. They are based on the principle first laid down by Charles Darwin of survival of the most fit. First pioneered by Holland [17], genetic algorithms have been widely studied, tested, and applied in many fields. A generic steady-state genetic algorithm is sketched in Fig. 1. The first step is the creation of a random population where each element is coded using a specific representation that encodes a set of features defined by the problem. Then, a *fitness function* is used to evaluate each individual, and the reproductive success varies with the fitness value. Two high-fitness elements are chosen for crossover and mutation. The procedure generates two new offspring that replace two random elements of the population. The process continues until the population's total fitness reaches a specified threshold or the number of genetic steps attains a predefined value.

In genetic algorithms, the *fitness function* of the problem leads the population to converge toward a population that fits the solution requirements. For complex problems, the definition of

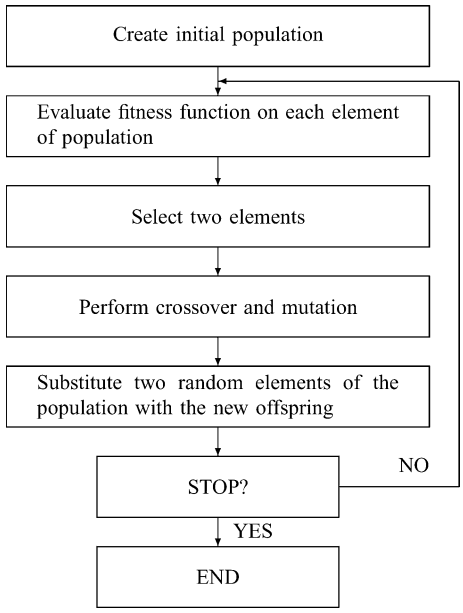


Fig. 1. A typical steady-state genetic algorithm.

an exact fitness function that describes perfectly the nature of the problem is often not possible and one is forced to use approximate fitness functions. This implies that during the selection procedure one cannot discriminate between two individuals with almost the same fitness value and a more fruitful approach is to select a fraction of high-fitness individuals and to use them for generating new offspring. This selection procedure is called *truncation selection* [18], [19]. In the *generational* approach, a new population is generated at every genetic step, which substitutes the old population. In the *incremental* (or *steady-state*) approach, only two new offspring are generated at every genetic step and inserted in the population. The latter approach is needed when we are dealing with varying fitness functions.

B. Introduction To Quantum Search Algorithms

The basic unit of information in quantum computation is the *qubit*. A qubit is a two-level quantum system and it can be represented by a unit vector of a two dimensional Hilbert space $(\alpha, \beta \in \mathbb{C})$

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \quad |\alpha|^2 + |\beta|^2 = 1$$

where we denote with $|0\rangle$ and $|1\rangle$ the basis states, adopting the *ket notation* for quantum state vectors. A two-level quantum system is described by a superposition of the basis states, whereas a two-level classical system can be just in one of the basis states 0 or 1.

The evolution of a quantum system is described by special linear operators, *unitary operators*[1] U which operate on qubits

$$U|\psi\rangle = U[\alpha|0\rangle + \beta|1\rangle] = \alpha U|0\rangle + \beta U|1\rangle.$$

An important consequence of the linearity of quantum operators is that the evolution of a two-level quantum system is the linear combination of the evolution of the basis states $|0\rangle$ and $|1\rangle$. This is known as *quantum parallelism*. On the contrary, in a two-level

[1]A linear operator is said to be *unitary* if $UU^\dagger = U^\dagger U = \mathbb{1}$, where $U^\dagger$ denotes the adjoint of the operator U.

classical system, we are forced to evolve the two possible states 0 and 1 separately. When we want to transfer information from the quantum system to a classical one, we have to perform *measurements* of the quantum state, whose result is probabilistic: we get the state $U|0\rangle$ with probability $|\alpha|^2$ and the state $U|1\rangle$ with probability $|\beta|^2$. The *no cloning theorem*, see [20], states that it is not possible to clone a quantum state $|\psi\rangle$ and, consequently, to obtain full information on the coefficients α and β from a single copy of $|\psi\rangle$. Another important feature arising from the linearity of quantum mechanics is *entanglement*. The state of a composite classical system AB is completely determined by the state of its subsystems. On the contrary, the state of a composite quantum system is the *tensor product* $\otimes$ of the states of the component systems; so a state of a composite system $|\psi\rangle_{\mathrm{AB}}$ could be like

$$|\mathrm{Bell}_{\mathrm{AB}} = \frac{1}{\sqrt{2}}[|0\rangle_{\mathrm{A}} \otimes |0\rangle_{\mathrm{B}} + |1\rangle_{\mathrm{A}} \otimes |1\rangle_{\mathrm{B}}]$$

which is not of the form $|\cdot\rangle_{\mathrm{A}} \otimes |\cdot\rangle_{\mathrm{B}}$. Such a *Bell state* is said to be *entangled*. Entanglement is a quantum resource that permits, for instance, quantum teleportation [16].

The two main quantum algorithms developed up to now are quantum Fourier transform (QFT) [1], and the Grover search algorithm [2]. QFT can be used to solve problems like discrete logarithm, order finding and factoring [22] and it lies out of the scope of this paper. The Grover algorithm has been used in the BBHT algorithm [23] (BBHT is the acronym of the authors' names) and in the Dürr–Høyer algorithm [16]. We briefly review the three algorithms next.

1) Grover Algorithm: The algorithm solves the problem of searching in an unstructured database. It has been shown that the Grover algorithm is $\mathcal{O}(\sqrt{N/t})$, where N is the number of entries in the database and t is the number of possible solutions [2]. Classical algorithms for solving this problem must, instead, look at each entry of the database until a solution is found, i.e., they are $\mathcal{O}(N/t)$. The basic idea of the Grover's algorithm is to amplify the coefficients of the superposition of all elements that correspond to the solutions of the given problem, while reducing the others. This procedure is performed by applying a unitary operator $\mathcal{O}(\sqrt{N/t})$ times. Then a measurement of the quantum state obtained will yield, with high probability, one of the possible solutions. The nonstructuredness requirement is essential for achieving the speedup stated above, otherwise, classical binary tree search would solve the problem in $\mathcal{O}(\log N)$. It should be emphasized that a classical procedure always permits to collect all the solutions in the database (by seeking through all the entries); on the contrary, the probabilistic nature of quantum measurement allows to get one solution at random among the solutions of the database. By repeating the whole quantum procedure, however, it is possible to obtain other solutions.

2) BBHT Algorithm: When the number of solutions is known in advance, one can use Grover's algorithm to look for one of them. Without previous knowledge of the number of solutions t marked by the oracle, one cannot use the Grover algorithm. This impossibility arises because in the amplitude amplification process we cannot compute the number of iterations to be performed in order to maximize the coefficients of the solution. However, when the number of solutions t is *a priori* unknown, it is still possible to use a remarkable quantum algorithm called BBHT [23] for finding a solution in a set of items $\{T_i\}_{i=0,\ldots,N-1}$ given an oracle that recognizes a solution.

Here, we give a brief summary of the BBHT algorithm and report the main complexity result. We assume, at first, that $1 \leq t \leq 3N/4$, where N is the total number of elements.

1) Initialize $m = 1$, set $\lambda = 6/5$ (any value between 1 and 4/3 would do) and create the state $|\Psi_0\rangle = H^{\otimes n}|0\rangle = (1/\sqrt{N})\sum_j |j\rangle$.
2) Choose i uniformly at random among the non-negative integers smaller than m.
3) Apply i iterations of Grover's algorithm starting from the initial state $|\Psi_0\rangle$.
4) Measure the register: let o be the outcome.
5) If the selected element T_o is a solution then `exit`.
6) Otherwise, set m to $\min(\lambda m, \sqrt{N})$ and go back to step 2.

The case $t > 3N/4$ can be treated in constant time by classical sampling.

Theorem 1.1: The BBHT algorithm finds a solution in an expected time of $\mathcal{O}(\sqrt{N/t})$.

Proof: See [23]. ∎

Remarks 1.2: As a step of the proof, the authors showed that the number of oracle queries is bounded from above by $4\sqrt{N/t} = \kappa_{\mathrm{BBHT}}\sqrt{N/t}$ when $t \ll N$.

3) Dürr–Høyer Algorithm: The Dürr–Høyer algorithm is a quantum algorithm for finding the minimum within an unsorted table of N items [16]. The core of the algorithm is a procedure which returns the index of an item smaller than the item determined by a particular threshold, by using the BBHT algorithm. This procedure is iterated until the minimum is reached. Dürr and Høyer showed that such an algorithm requires an expected number of $\mathcal{O}(\sqrt{N})$ iterations.

4) Quantum Evaluation of Functions: In classical computation, a small set of classical gates (e.g., AND OR NOT) can be used to compute an arbitrary classical function; a similar result is still true in quantum computation.

A set of gates is said to be *universal for quantum computation* if any unitary operation may be approximated to arbitrary accuracy by a quantum circuit involving only those gates. It has been shown that using *Hadamard, phase*, CNOT, and $\pi/8$ gates, any arbitrary unitary operation can be approximated to arbitrary accuracy [22] Moreover, any classical circuit can be made reversible by introducing a special gate named *Toffoli gate*. The Toffoli gate has three input bits, a, b, and c; a and b are the first and the second "control bits," while c is the "target bit." The gate does not change the control bits and flips the target bit only if both control bits are set. The Toffoli gate can be used to implement NAND and FANOUT and it is reversible. Since a quantum version of the Toffoli gate has been developed (see, e.g., [22]), a classical reversible circuit that computes a function $f : \{0,1\}^n \to \{0,1\}^m$ can be converted to a quantum circuit that computes the same function. Note that if the function is not injective, one can use ancilla qubit to make the circuit reversible.

II. Quantum Genetic Optimization Algorithm (QGOA)

The basic structure of our QGOA is based on the classical structure of a steady-state genetic algorithm. We present here the problem using a "global search" strategy, where we are considering all the elements of the population. In particular, we

CLASSICAL GENETIC ALGORITHM

Given a representation of the population and the fitness function
Repeat M **times**

- **Evaluate fitness**: Evaluate the fitness of every element of the population.
- **Select two elements**: Select a subpopulation using *truncation selection* (a fraction p of the best elements of the population) and then choose randomly two elements from it.
- **Crossover and mutation**: Perform crossover of the two elements by exchanging two random substrings. Then with probability P_{M} mutate each allele (i. e. bit) of the strings.
- **Substitution**: Choose two random elements from the population and then replace them with the new offspring.

QUANTUM GENETIC OPTIMIZATION ALGORITHM

Given a qubit representation of the population and a quantum evaluation unit
Repeat M **times**

- **Select two elements [Quantum]**: Use the *quantum selection procedure* (which performs the creation of a superposition of all elements of the population and the application of the quantum fitness evaluation unit) to choose one element. Run it again to choose another element.
- **Crossover and mutation [Classical]**: As above.
- **Substitution [Classical]**: As above.

Fig. 2. Classical and QGOA. Note that we need to run the quantum selection procedure twice because the measurement process destroys the superposition of the elements.

have developed a quantum selection procedure that includes a quantum fitness evaluation unit.

In Fig. 2(a), comparison between the classical genetic algorithm and the QGOA is shown. Notice that no external quantum evaluation procedure is needed since quantum fitness recalculation is computed inside the quantum selection procedure. This procedure is based on the quantum algorithm for finding the minimum proposed by [16], where it was shown that it is possible to find the minimum of a list by using a variant of the Grover quantum search algorithm in $\mathcal{O}(\sqrt{N})$.

By reducing the number of iterations, we show that we can select a subpopulation of optimal elements in constant time and that the convergence speed, in terms of genetic steps, of such an algorithm is comparable to the convergence speed of a classical steady-state genetic algorithm with truncation selection. The main difference is that, in the quantum selection procedure, at each genetic step the choice of an optimal subpopulation is performed in constant time, whereas in a classical selection procedure an ordering algorithm is needed.

A. Quantum Fitness Evaluation Unit

As explained in the introduction, given a classical reversible circuit that computes a fitness function $F(j) = F_j$, where $j \in \{0, \ldots, N-1\}$ are the elements of the population in binary representation, it can be converted into a quantum circuit yielding a *quantum fitness evaluation* operator U_F. Clearly, there is no general recipe for constructing U_F because its physical realization depends on the problem at hand.

If we use quantum binary encodings[2] for the elements, the superposition of all elements of the population is denoted by

$$|\Psi\rangle = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} |j\rangle$$

and the action of the quantum black box results in

$$U_F|\Psi\rangle|0\rangle = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} |j\rangle|F_j\rangle.$$

[2]Given $j = b_0 * 2^0 + b_1 * 2^1 + \cdots + b_{n-1} * 2^{n-1}$, where $b_i \in \{0, 1\}$, then $|j\rangle = |b_0\rangle \otimes |b_1\rangle \otimes \cdots \otimes |b_{n-1}\rangle$

Hence, using U_F only once, we can compute all the fitness values $\{F_j \,|\, j = 0, \ldots, N-1\}$ of the population, whereas the classical procedure requires N fitness evaluations. The process of measurement would destroy such a superposition, giving us only one fitness value. So at this stage, we could not gain any useful information on the best elements of the population. The oracle of the quantum selection procedure includes this unit to "mark" all the elements of the population that fulfill the condition $F_j \geq F_y$, where y is a threshold index.

The oracle is always the same during the computation. Its input is a superposition of all the N elements of the population at every genetic step (this is a "global search"). Hence, its capacity is N. The oracle is the same at every genetic step; however, the fitness function can vary between steps depending on some external time-dependent physical input, in addition to the logical input provided by the qubits.

B. Quantum Selection Procedure

The quantum selection procedure is based on the algorithm for finding the minimum of a list of N items [16]. The authors showed that for finding the (absolute) minimum, a number of iterations $\mathcal{O}(\sqrt{N})$ is needed. Here, we are not interested in finding the minimum, but in selecting a subpopulation of near-optimal elements of the whole population, namely, elements with a relatively high value of fitness. The algorithm works as described in Fig. 3.

Defination 2.1: A *Dürr–Høyer iteration* is the sequence of operations defined in 2a, 2b, and 2c of the quantum selection algorithm. We denote with n_{h} the number of Dürr–Høyer iterations.

Remarks 2.2: When $n_{\mathrm{h}} = 1$, we obtain the BBHT algorithm. Dürr and Høyer analyzed the case $n_{\mathrm{h}} = \infty$.

One might argue that a probabilistic algorithm could do about the same, by choosing $\mathcal{O}(\log R)$ elements, where R is a fraction of the entire population, evaluating the fitness function for the chosen elements (and only for them), and picking the best one. Such an assumption is not correct since the convergence of the genetic algorithm is different for the two selection procedures. After n_{h} iterations, we have a probability for choosing the best element of the population equal to R/N; instead, in this

QUANTUM SELECTION PROCEDURE

1) Choose randomly an index $y \in \{0, 1, \ldots, N-1\}$ corresponding to the threshold F_y. Compute classically $F_y = F(y)$.
2) Perform n_{h} times:
   a) Initialize memory to $|0\rangle|y\rangle$.
   b) Perform the algorithm BBHT (the step 1 of BBHT transforms the state $|0\rangle|y\rangle$ into $\frac{1}{\sqrt{N}}\sum_j |j\rangle|y\rangle$), where the oracle (that includes a quantum fitness evaluation unit) inverts the amplitude of the elements that satisfy $F_j \geq F_y$.
   c) Measure the first ket and get a new index y'. Compute classically $F_{y'} = F(y')$. If $F_{y'} > F_y$ then set the index y to y'.
3) Return the index y.

Fig. 3. The quantum selection algorithm.

classical probabilistic algorithm, the probability would be only $\log R/N$ (exponentially smaller). The main difference is that in one case we choose among the best elements, in the other, we choose in a completely random way.

III. COMPLEXITY OF THE ALGORITHM

In this section, we present a complexity analysis of the QGOA, in order to compare it with a classical genetic algorithm. We do not consider the computational cost of crossover, mutation, and substitution of the QGOA because they are constant for each genetic step and classical for both algorithms, and concentrate our analysis on the quantum selection procedure, whose time-complexity in terms of oracle calls will be deeply investigated. The time required for a single oracle call will depend on the technology used for implementing the oracle.

Let us consider the complexity of the quantum selection procedure step by step. Steps 1) and 3) of the quantum selection procedure do not enter in the complexity calculation since they are performed only once and in constant time. Step 2a) initializes the quantum memory and it is performed n_{h} times. Step 2c) performs the measurement process and it requires n_{h} classical computations of the fitness function. Step 2b), in terms of number of n-qubit operators, is the most onerous and, from the point of view of the complexity, it requires a deeper analysis. We will analyze this step in terms of number of *oracle calls*. The oracle includes the quantum fitness evaluation unit and inverts the amplitude of the elements with fitness greater than or equal to a given threshold F_y. We will consider an oracle call as the *time step unit* for our analysis of step 2b) without taking into account steps 1), 3), 2a), and 2c), because their cost depends linearly on n_{h} and it does not depend on the number of qubits $n = \log N$.

We are interested in the expected number of oracle calls in the quantum selection procedure; it is known that the BBHT algorithm requires $\mathcal{O}(\sqrt{N/t})$ oracle calls, where t is the number of marked elements (see Theorem 1.1). Dürr and Høyer found that the expected number of oracle calls of their algorithm in order to find the minimum is $22.5\sqrt{N}$. In our algorithm, the number of Dürr–Høyer iterations is a parameter, and we need to characterize its relation with the expected number of oracle calls. We will show in this section (Theorem 3.4) that the expected number of oracle calls is bounded from above by $\kappa \cdot 2(2^{n_{\mathrm{h}}} - 1)$, where κ is a constant and n_{h} is the number of Dürr–Høyer iterations. This is our main result because it states that the expected number of oracle calls does not depend on the dimension of the population N. In order to show this result, we will need a bound on the expected number of oracle calls (Theorem 3.3). Moreover, we will show that n_{h} is directly related to the selection pressure (Theorem 3.7).

In order to characterize the expected number of oracle calls of step 2b) of the *quantum selection procedure*, we need to prove a Lemma. We consider a list of N elements and a fitness function f that maps each element onto a real positive value.

We define as the *rank* of an element its position $s \in \{1, N\}$ in the list sorted in descending order of fitness function values.

Lemma 3.1: The probability $\Pr(s, m)$ of choosing an element of rank s as threshold before the mth Dürr–Høyer iteration, as shown in (1) at the bottom of the page, where $\theta(x)$ is the step function.[3]

Proof: We denote with $\Pr(s, l)$ the probability that we choose an element of rank s before the lth Dürr–Høyer iteration and with $\Pr(s \,|\, j, l)$ the conditional probability that we choose an element of rank s before the lth Dürr–Høyer iteration, after an element of rank j has been chosen in the previous iteration. We use the *total probability* equation

$$\Pr(s, l) = \sum_{j=1}^{N} \Pr(s \,|\, j, l) \cdot \Pr(j, l-1)$$

which holds because the set of possible events "choosing an element of rank j" is a partition of the set of events. During

[3]
$$\theta(x) = \begin{cases} 1, & x \geq 0 \\ 0, & \text{otherwise} \end{cases}.$$

$$\Pr(s, m) = \begin{cases} \frac{1}{N}, & \text{if } m = 1 \\ \sum_{j_m=1,\, j_{m-1}=1,\ldots,\, j_2=1}^{N} \frac{\theta(j_m - s)\theta(j_{m-1} - j_m)\cdots\theta(j_2 - j_3)}{j_m \cdots j_{m-1} \cdots j_3 \cdot j_2 \cdot N} & \end{cases} \tag{1}$$

each Dürr–Høyer iteration, we have that $\Pr(s|j,l) = (1/j)$ if $s \leq j$ or zero, otherwise, as ensured by step 3 of the algorithm

$$\Pr(s|j,l) = \frac{\theta(j-s)}{j}.$$

The first index is chosen uniformly at random from all elements, so $\Pr(s,1) = (1/N)$, where $N = 2^n$. Using the *total probability* equation recursively, we finally obtain the first equation at the bottom of the page.

Definition 3.2: Let $\mathcal{N}_m$ be the random variable *number of oracle calls during the mth Dürr–Høyer iteration*. Moreover, let $\mathcal{N}$ be the random variable *total number of oracle calls in the quantum selection procedure*.

The following theorem uses the previous Lemma in order to bound the expected number of oracle calls.

Theorem 3.3: The expectation of the total number of oracle calls in the quantum selection procedure is

$$\mathbb{E}[\mathcal{N}] \leq \frac{\kappa}{\sqrt{N}} \sum_{s=1}^{N} \frac{1}{\sqrt{s}} \cdot Q(s) \tag{2}$$

where $Q(s)$ is defined in the second equation at the bottom of the page and κ is a constant.

Proof: From the very definition of expectation and Theorem 1.1, the expected number of oracle calls during the mth Dürr–Høyer iteration is

$$\mathbb{E}[\mathcal{N}_m] \leq \sum_{s=1}^{N} \kappa \sqrt{\frac{N}{s}} \cdot \Pr(s,m)$$

using Lemma 3.1 (1), we show equation (3) at the bottom of the page. From the definition of $\mathcal{N}$, it is clear that

$$[\mathcal{N}] = \sum_{m=1}^{n_{\mathrm{h}}} \mathcal{N}_m$$

whence we obtain

$$\mathbb{E}[\mathcal{N}] \leq \sum_{m=1}^{n_{\mathrm{h}}} \mathbb{E}\mathcal{N}_m.$$

■

The bound of Theorem 3.3 depends on the number of elements of the population. We now want to calculate another upper bound for the expectation of $\mathcal{N}$. In particular, this upper bound is independent of the cardinality of the population, as stated by the following theorem.

$$\begin{aligned}
\Pr(s,2) &= \sum_{j_2=1}^{N} \Pr(s \,|\, j_2, 2) \cdot \Pr(j_2, 1) \\
&= \sum_{j_2=1}^{N} \theta(j_2 - s) \frac{1}{j_2 \cdot N} \\
\Pr(s,3) &= \sum_{j_3=1}^{N} \Pr(s \,|\, j_3, 3) \cdot \Pr(j_3, 2) \\
&= \sum_{j_3=1}^{N} \sum_{j_2=1}^{N} \frac{\theta(j_3 - s)\theta(j_2 - j_3)}{j_3 \cdot j_2 \cdot N} \\
&\vdots \\
\Pr(s,m) &= \sum_{j_m=1,\, j_{m-1}=1,\ldots,j_2=1}^{N} \frac{\theta(j_m - s)\theta(j_{m-1} - j_m) \cdots \theta(j_2 - j_3)}{j_m \cdot j_{m-1} \cdots j_3 \cdot j_2 \cdot N}
\end{aligned}$$

■

$$Q(s) = 1 + \sum_{m=2}^{n_{\mathrm{h}}} \sum_{j_m=1,\, j_{m-1}=1,\ldots,j_2=1}^{N} \frac{\theta(j_m - s)\theta(j_{m-1} - j_m) \cdots \theta(j_2 - j_3)}{j_m \cdot j_{m-1} \cdots j_3 \cdot j_2}$$

$$\mathbb{E}[\mathcal{N}_m] \leq \begin{cases} \frac{\kappa}{\sqrt{N}} \sum_{s=1}^{N} \frac{1}{\sqrt{s}} & \text{if } m = 1 \\ \frac{\kappa}{\sqrt{N}} \sum_{s=1,\, j_m=1,\, j_{m-1}=1,\ldots,j_2=1}^{N} & \frac{\theta(j_m - s)}{\sqrt{s}} \left[\prod_{l=3}^{m} \frac{\theta(j_{l-1} - j_l)}{j_l} \cdot \frac{1}{j_2} \right] \end{cases} \tag{3}$$

Theorem 3.4: The expected number of oracle calls in the quantum selection procedure is bounded by

$$\mathbb{E}[\mathcal{N}] < \kappa \cdot 2(2^{n_\mathrm{h}} - 1) \tag{4}$$

where κ is a decreasing function of n_h.

Proof: First, we show that for all $m \in \{1, N\}$

$$\mathbb{E}[\mathcal{N}_m] < \kappa \cdot 2^m. \tag{5}$$

From calculus, we have that

$$\sum_{s=1}^{N} \frac{1}{\sqrt{s}} < 1 + \int_1^N \frac{1}{\sqrt{s}}\,\mathrm{d}s = 2\sqrt{N} - 1 < 2\sqrt{N}.$$

Taking $\kappa = 1$ for simplicity of notation, for $m = 1$, from (3) it follows:

$$\frac{1}{\sqrt{N}} \sum_{s=1}^{N} \frac{1}{\sqrt{s}} < \frac{1}{\sqrt{N}}(2\sqrt{N} - 1) < 2.$$

For $m > 1$, we have the equation shown at the bottom of the page. Now, using Theorem 3.3 and the above results

$$\mathbb{E}[\mathcal{N}] < \kappa \sum_{m=1}^{n_\mathrm{h}} 2^m = \kappa \cdot 2(2^{n_\mathrm{h}} - 1).$$

■

Remarks 3.5: It is important to emphasize that the bound depends on n_h only and does not depend on the dimension of the population N.

We have seen that the number of Dürr–Høyer iterations n_h determines an upper bound to the number of oracle calls during the quantum selection; it is an important parameter of our algorithm and we want to understand deeply its meaning.

Defination 3.6: We denote with $\mathcal{T}_m$ the random variable *number of marked elements after the mth Dürr–Høyer iteration*.

Theorem 3.7: Let $N = 2^n$; the expected number of marked elements after m Dürr–Høyer iterations is

$$\mathbb{E}[\mathcal{T}_m] = 1 + (2^n - 1) \cdot 2^{-m}. \tag{6}$$

Proof: For $m = 1$, $\mathbb{E}[\mathcal{T}_1] = \sum_{s=1}^{N} s \cdot \Pr(s, 1) = (N + 1)/2$. For $m > 1$, we change the order of summation and obtain the equation shown at the bottom of the next page. With $m = n_\mathrm{h}$, Theorem 3.7 shows clearly how n_h determines the expected number of marked elements, and thus the selection pressure. The effect of Dürr–Høyer iterations is shown in Fig. 4. The cardinality of the marked subpopulation approximatively halves for increasing n_h. This implies that n_h grows logarithmically with the number of marked elements.

IV. Simulation

In this section, we present the results of a simulation of the QGOA in order to show the validity of Theorem 3.3 and Theorem 3.7 which bound the expected number of oracle calls and characterize the selection pressure, respectively. We used a particular fitness function in order to compare the convergence speed of the total fitness of the QGOA with respect to a classical genetic algorithm with truncation selection.

Simulations of the classical genetic algorithm and of the quantum genetic algorithm were performed using the symbolic language Mathematica™. The quantum fitness evaluation unit was simulated as a black box without modeling the quantum circuits. The maximum number of qubits used is $n = 8$, because beyond that value too many computational resources were needed, since the resources needed to simulate a quantum computer on a classical one increase exponentially with n.

A. Fitness Function

The fitness value of each element of the population reflects the quality of the characteristics that it encodes. It is quite common

$$\begin{aligned}
&\sum_{s=1}^{N} \sqrt{\frac{N}{s}} \cdot \sum_{j_m=1,\, j_{m-1}=1,\ldots,j_2=1}^{N} \frac{\theta(j_m - s)\theta(j_{m-1} - j_m)\cdots\theta(j_2 - j_3)}{j_m \cdot j_{m-1} \cdots j_3 \cdot j_2 \cdot N} \\
&\quad = \frac{1}{\sqrt{N}} \sum_{j_2=1}^{N} \sum_{j_3=1}^{j_2} \cdots \sum_{j_m=1}^{j_{m-1}} \sum_{s=1}^{j_m} \frac{1}{\sqrt{s}} \cdot \frac{1}{j_m \cdot j_{m-1} \cdots j_3 \cdot j_2} \\
&\quad < \frac{1}{\sqrt{N}} \sum_{j_2=1}^{N} \sum_{j_3=1}^{j_2} \cdots \sum_{j_m=1}^{j_{m-1}} 2\sqrt{j_m} \cdot \frac{1}{j_m \cdot j_{m-1} \cdots j_3 \cdot j_2} \\
&\quad = \frac{2}{\sqrt{N}} \sum_{j_2=1}^{N} \sum_{j_3=1}^{j_2} \cdots \sum_{j_m=1}^{j_{m-1}} \frac{1}{\sqrt{j_m}} \cdot \frac{1}{j_{m-1} \cdots j_3 \cdot j_2} \\
&\quad < \frac{2^2}{\sqrt{N}} \sum_{j_2=1}^{N} \sum_{j_3=1}^{j_2} \cdots \sum_{j_{m-1}=1}^{j_{m-2}} \sqrt{j_{m-1}} \cdot \frac{1}{j_{m-1} \cdots j_3 \cdot j_2} \\
&\quad \vdots \\
&\quad < \frac{2^{m-1}}{\sqrt{N}} \sum_{j_2=1}^{N} \frac{1}{\sqrt{j_2}} < \frac{2^{m-1}}{\sqrt{N}} \cdot 2\sqrt{N} = 2^m
\end{aligned}$$

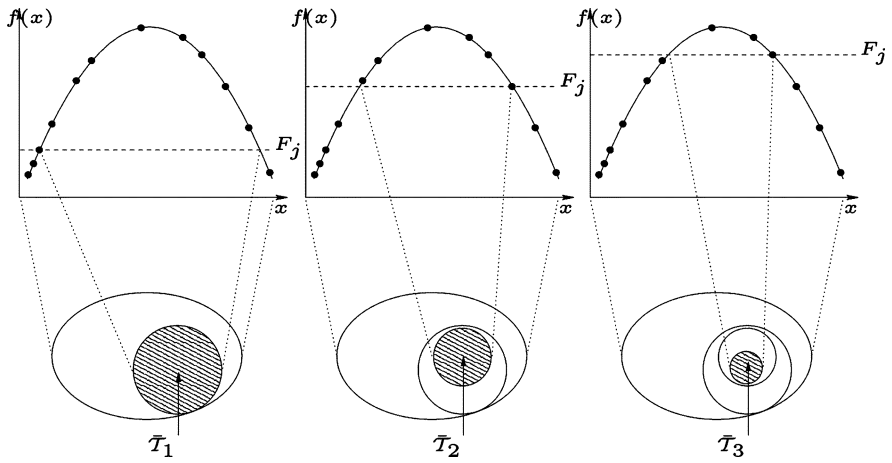


Fig. 4. Change in the cardinality of the subpopulation when n_{h} is changed from 1 to 3.

to have a noisy environment in which the problem is being studied, which means that the fitness function can vary at every genetic step. We refer to the class of fitness functions which can vary at every genetic step as *varying fitness functions*. We have simulated a varying function by adding to the fitness value of an element a random quantity ϵ obtained from a Gaussian distribution of mean value 0 and variance $\sigma_\epsilon = 10 \cdot \mu$, where μ is the mutation probability.[4] The multipeak varying fitness function used in the simulations is

$$f(x) = \sin(\pi x) \cdot (9x \bmod 1) + \epsilon_{\mathrm{Gaussian}}(0, \sigma_\epsilon). \quad (7)$$

[4]If the value of σ_ϵ is too small, the mutation procedure masks the noisy effect of the noisy fitness function.

$$\begin{aligned}
\mathbb{E}[\mathcal{T}_m] &= \sum_{s=1}^{N} s \cdot \Pr(s,m) \\
&= \sum_{s=1}^{N} s \cdot \sum_{j_m=1,\, j_{m-1}=1,\ldots,j_2=1}^{N} \frac{\theta(j_m - s)\theta(j_{m-1} - j_m)\cdots\theta(j_2 - j_3)}{j_m \cdot j_{m-1}\cdots j_3 \cdot j_2 \cdot N} \\
&= \sum_{j_2=1}^{N}\sum_{j_3=1}^{j_2}\cdots\sum_{j_m=1}^{j_{m-1}}\sum_{s=1}^{j_m} s \cdot \frac{1}{j_m \cdot j_{m-1}\cdots j_3 \cdot j_2 \cdot N} \\
&= \sum_{j_2=1}^{N}\sum_{j_3=1}^{j_2}\cdots\sum_{j_m=1}^{j_{m-1}} \frac{j_m(j_m+1)}{2} \cdot \frac{1}{j_m \cdot j_{m-1}\cdots j_3 \cdot j_2 \cdot N} \\
&= \frac{1}{2\cdot N}\sum_{j_2=1}^{N}\sum_{j_3=1}^{j_2}\cdots\sum_{j_m=1}^{j_{m-1}} \frac{j_m+1}{j_{m-1}\cdots j_3 \cdot j_2} \\
&= \frac{1}{2\cdot N}\sum_{j_2=1}^{N}\sum_{j_3=1}^{j_2}\cdots\sum_{j_{m-1}=1}^{j_{m-2}} \cdot \left(\frac{j_{m-1}(j_{m-1}+1)}{2} + j_{m-1}\right) \cdot \\
&\quad \cdot \frac{1}{j_{m-1}\cdots j_3 \cdot j_2} \\
&= \frac{1}{4\cdot N}\sum_{j_2=1}^{N}\sum_{j_3=1}^{j_2}\sum_{j_{m-1}=1}^{j_{m-2}} \frac{j_{m-1}+3}{j_{m-2}\cdots j_3 \cdot j_2} \\
&\vdots \\
&= \frac{1}{2^{m-1}\cdot N}\sum_{j_2=1}^{N}(j_2 + 2^{m-1} - 1) \\
&= \frac{1}{2^{m-1}\cdot N}\left(\frac{N(N+1)}{2} + N\cdot(2^{m-1}-1)\right) \\
&= \frac{N + 2^m - 1}{2^m}
\end{aligned}$$

■

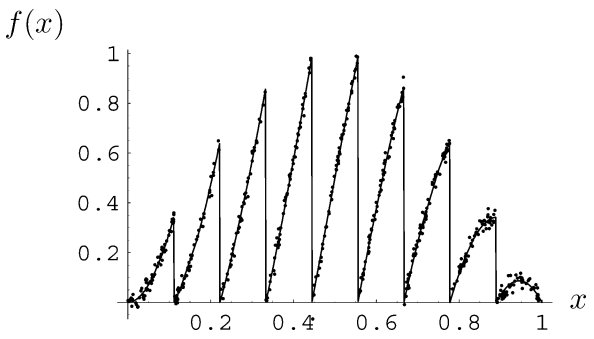


Fig. 5. Main fitness function used in the simulation. A realization of the added Gaussian noise is also shown.

TABLE II
MEAN NUMBER OF ORACLE CALLS AND ITS STANDARD DEVIATION IN THE QUANTUM SELECTION PROCEDURE AS RESULTS OF THE SIMULATIONS

n	$n_{\rm h}=1$	$n_{\rm h}=2$	$n_{\rm h}=3$	$n_{\rm h}=4$
2	6.3 ± 1.3	10.9 ± 1.6	-	-
3	6.4 ± 0.9	10.6 ± 1.3	13.4 ± 1.2	-
4	6.8 ± 0.8	10.8 ± 0.8	14.4 ± 0.8	18.2 ± 1.0
5	6.6 ± 0.4	11.2 ± 0.4	15.6 ± 0.6	21.0 ± 0.8
6	6.8 ± 0.3	11.7 ± 0.3	17.6 ± 0.4	25.1 ± 0.6

TABLE III
REGRESSION COEFFICIENTS OF THE (2) DATA IN TABLE II

$n_{\rm h}$	Coefficient $\kappa_{\rm reg}$	Coefficient of determination R^2
1	3.79 ± 0.08	0.9984
2	2.52 ± 0.07	0.9965
3	2.00 ± 0.03	0.9989
4	1.76 ± 0.02	0.9997

This function is plotted in Fig. 5. Notice that even if the function in not injective, it is possible to build a reversible circuit for computing such function (as discussed in the introduction).

B. Expected Number of Oracle Calls

Equation (2) gives a bound on the expected number of oracle calls in the quantum selection procedure. We recall that we need two elements of the population to cross over, so we have to run the quantum selection algorithm twice (or more if the elements coincide) to obtain two different elements because the measurement process destroys the quantum superposition. We can argue that for a large population it suffices to run it only twice.

To verify (2) (notice that $Q(s)$ depends on $n_{\rm h}$, which is a parameter of the QGOA), we considered different population cardinalities $N=2^n$ with $n=2,3,4,5,6$. For each population cardinality, we have generated 100 random populations, and for different values of $n_{\rm h}=1,2,3,4$, we have computed the mean number of oracle calls in the quantum selection procedure.[5] Results are shown in Table II.[6] In order to verify the bound, we need an estimate of the constant κ appearing in (2). Unfortunately, an estimate is known only for $n_{\rm h}=1$ and $t\ll N$ (BBHT algorithm). Our strategy was to fit the bound against the data and to compare the values of the parameters. Then, we ran a regression on the experimental points using (2) and estimated κ, as shown in Table III. Finally, Fig. 6 shows the experimental plots (and error bars) and the regression function for different numbers of Dürr–Høyer iterations.

[5]Theorem 3.7 shows clearly how $n_{\rm h}$ determines the expected number of marked elements, and thus the selection pressure (see Fig. 4).

[6]Some combinations of n and $n_{\rm h}$ are useless because the quantum selection procedure selects almost always the element with maximum fitness, hence, it is not possible to cross over two different elements.

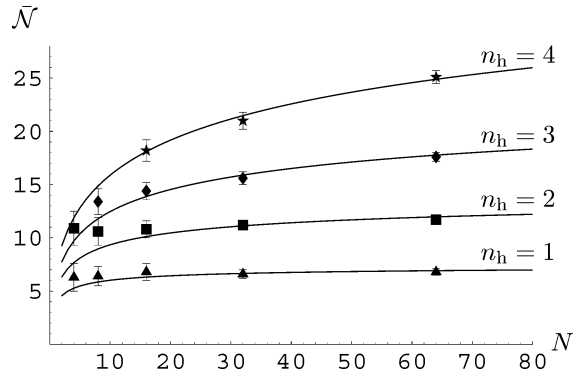


Fig. 6. Mean number of oracle calls for different values of the number of elements of the population N and with numbers of Dürr–Hoyer iterations $n_{\rm h}=1,2,3,4$. Experimental data and fitted curves based on Theorem 3.4.

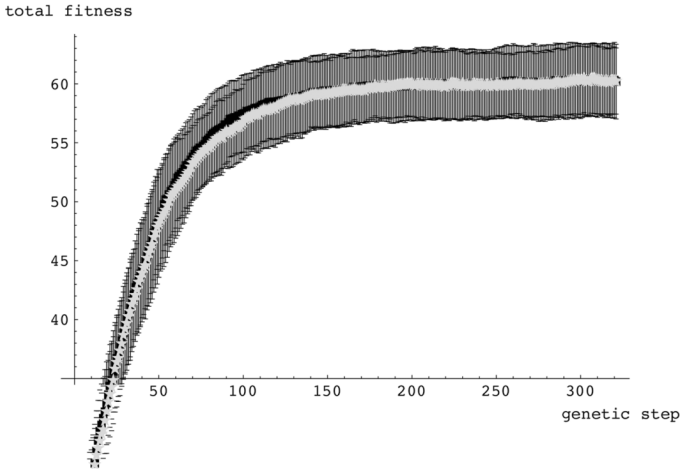


Fig. 7. Total fitness (mean and variance) of quantum genetic optimization algorithm (brighter line) and classical genetic algorithm with truncation selection as a function of the number of genetic steps. Each genetic step requires $\mathcal{O}(N\log N)$ in the classical selection procedure and $\mathcal{O}(1)$ in the quantum selection procedure.

When $n_{\rm h}=1$, our quantum selection procedure coincides with BBHT (Remark 2.2), so it is interesting to compare the empirical value with the theoretical bound. From Remark 1.2, $\kappa_{\rm BBHT}\approx 4$. In Table III, we obtain $\kappa_{\rm reg}=3.79\pm0.08$ for $n_{\rm h}=1$. However, since in the selection procedure we need two different elements to crossover, we expect to use the quantum selection procedure at least twice. Hence, $\kappa\leq 3.79/2=1.895<4=\kappa_{\rm BBHT}$.

C. Performance Comparison

Here, we show that the convergence speed, in terms of genetic steps, of the QGOA is comparable to a classical truncation selection algorithm, where two elements of the fraction of the population are used to generate the new offspring. This means that the total fitness function (the sum of all fitness values of the elements of the population) versus genetic steps should be equal within the statistical errors. The real power of the QGOA is exploited at each genetic step, where the computational complexity of the fitness selection procedure is $\mathcal{O}(1)$.

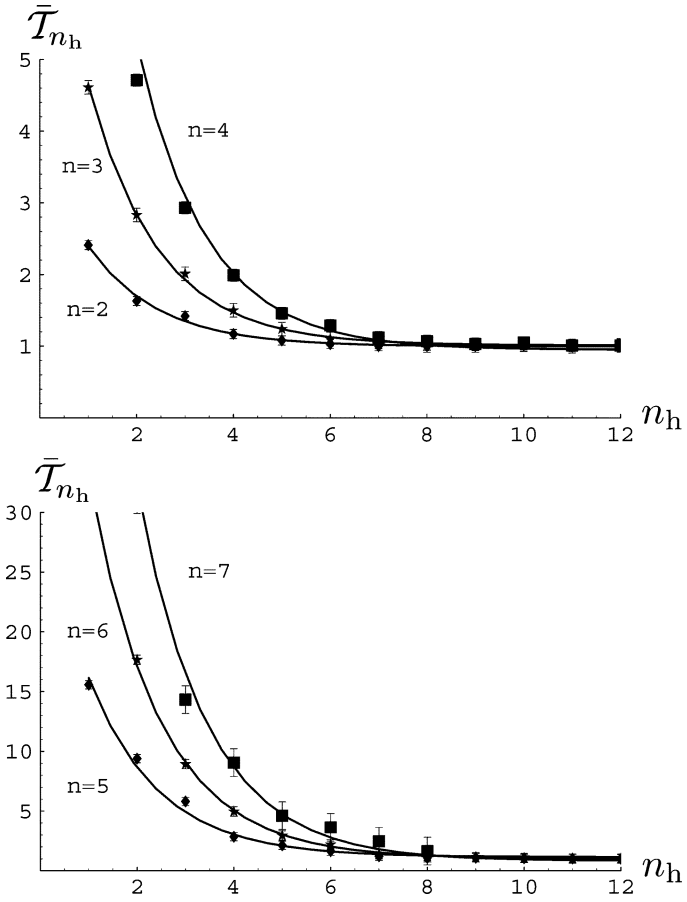


Fig. 8. Mean number of marked elements for different values of the number of Dürr–Høyer iterations. Experimental data and regression fitted curves based on Theorem 3.7.

TABLE IV
REGRESSION OF SIMULATION DATA

n	$\mathbb{E}[\mathcal{T}_{n_h}]$	R^2
2	$(1.01 \pm 0.02) + (2.81 \pm 0.12) \cdot 2^{-n_h}$	0.9816
3	$(0.98 \pm 0.03) + (7.76 \pm 0.19) \cdot 2^{-n_h}$	0.9940
4	$(0.95 \pm 0.03) + (16.16 \pm 0.17) \cdot 2^{-n_h}$	0.9989
5	$(1.06 \pm 0.11) + (30.53 \pm 0.69) \cdot 2^{-n_h}$	0.9944
6	$(1.19 \pm 0.13) + (64.46 \pm 0.78) \cdot 2^{-n_h}$	0.9986
7	$(0.96 \pm 0.52) + (125.81 \pm 2.53) \cdot 2^{-n_h}$	0.9976

The number of genetic steps performed during the simulation is a multiple of $N/2$. After $N/2$ genetic steps, we expect on average a complete change of the population (namely, a new generation). Hence, after M genetic steps, we expect a number of generations $I \approx 2M/N$. The simulation has been performed using the same fitness function of (7) and with $I = 10$. The results of a simulation with a population of cardinality $64 = 2^6$ and $n_h = 3$ (i.e., about a fraction of $1/8$ of the population at each genetic step) are shown in Fig. 7 and they confirm the analysis made.[7] Finally, the regressions for the mean number of marked solution as a function of n_h and for different values of n are shown in Table IV; the corresponding plot is shown in Fig. 8.

[7]We have done other simulations by changing the number of qubits and n_h, we obtain that the two curves are the same within the errors. See [24].

V. CONCLUSION

When the first quantum computers will start becoming available for applications, the need for quantum algorithms exploiting the power of such hardware will be pressing and the existing quantum algorithms will be subject to test. The number of quantum algorithms that fully exploit the power of quantum computation in order to gain significant speedup is rather limited. Hence, a general approach for applying quantum computation to a wide range of problems is needed.

Our efforts in a such direction have yielded a QGOA that outperforms its classical analogue in terms of number of oracle calls. However, as explained previously, starting from the complexity of Grover's algorithm, we know that we can speed up the process only if no structure is defined on the problem (hence, the name "unstructured database search" used to refer to the Grover quantum search algorithm).[8] Such a requirement implies that, in order to achieve a quantum speedup, we must restrict the problem class to varying fitness functions, where the structure created by the evaluation of population elements is "broken" at every genetic step. In other words, *in order to gain a significant advantage over a classical approach using a quantum algorithm based on Grover search algorithm, we have to consider problems where the fitness function is varying*.

Under these conditions, our QGOA outperforms the classical one in terms of oracle calls. In fact, whereas the classical selection procedure requires $\mathcal{O}(N \log N)$ for reordering of the elements, we have shown that the quantum selection procedure requires only $\mathcal{O}(1)$ quantum oracle calls. Our results do not contradict the well-known fact that in the black-box model the quantum speedup can be at most polynomial in the number of qubits. In fact, our algorithm does not search for a single marked element but for a fraction of marked elements with high fitness.

The quantum fitness evaluation unit has to be implemented inside the quantum selection procedure, which is performed twice, and it computes the fitness in parallel on a superposition of elements at every genetic step. On the contrary, a classical fitness evaluation has to be performed N times at every genetic step. We have to note that the quantum selection procedure selects the best elements of the population (the selection pressure depends on a parameter of the QGOA, n_h), and from them two elements are randomly chosen for the mating pool.

Truncation selection is one of the selection procedures used in classical genetic algorithms. It computes the fitness values of all the elements of the population, it orders them accordingly, and it picks randomly two or more elements among a fraction of the best ones.

The convergence speed of our algorithm, in terms of genetic steps, is comparable to a classical genetic algorithm with truncation selection, and the real power of quantum computation is exploited at every genetic step, where the fitness evaluation and selection procedure are performed in $\mathcal{O}(1)$. Moreover, the selection pressure of the algorithm can be controlled by a parameter of the QGOA n_h.

QGOA is a quantum algorithm that combines the principles of genetic computation with the principles of quantum search.

[8]If the fitness function is fixed a structure can be created by ordering the initial results of fitness computation in $\mathcal{O}(N \log N)$ and maintaining the order in $\mathcal{O}(\log N)$, exploiting such informations to speed up the computation.

The result is that running on a quantum machine QGOA will provide a sensible speedup from $\mathcal{O}(N \log N)$ to $\mathcal{O}(1)$ on each genetic step, where N is the dimension of the population. This result permits the use of bigger populations as the number of qubits $(\log N)$ used for the encoding will hopefully grow thanks to technology. The advantage will be far more useful for varying fitness function, for example, in quantum control processes. In this case, each oracle interrogation is affected via an instant realization of the process involved and, therefore, it is affected by unavoidable imperfections and noise, as no real laboratory experiment can be performed with ideally perfect conditions. Thus, the physical "black box" embodying the oracle remains the same and does not need to be rebuilt at every step; nevertheless, the value of the fitness function (in our example, the probability amplitude to reach a desired final state as a result of the quantum chemical reaction) is subject to fluctuations from step to step. This is relevant to quantum computation in general, beyond the specific example outlined here, as in that context one can never fully disregard the physical embodiment of the logical operations.

In this sense, a genetic algorithm (like QGOA) that works in the presence of noise can be regarded as an example of built-in algorithmic fault tolerance, and this is a major advantage with respect to its classical counterpart, as we have demonstrated quantitatively in our work.

When and how a quantum machine will be available is an open question. However, our proposal will permit to apply the advantages of quantum computation to a broader set of problems related to genetic algorithms.

REFERENCES

[1] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proc. 35th Annu. Symp. Found. Comput. Sci.*, Los Alamitos, CA, Nov. 1994, pp. 124–134.

[2] L. K. Grover, "Quantum mechanics helps in searching for a needle in a haystack," *Phys. Rev. Lett*, vol. 79, no. 2, pp. 325–328, Jul. 1997.

[3] D. E. Goldberg, *Genetic algorithms in search, optimization, and machine learning.*. Reading, MA: Addison-Wesley, 1989.

[4] C. Aporntewan and P. Chongstitvatana, "A hardware implementation of the compact genetic algorithm," in *Proc. 2001 Congr. Evol. Comput. CEC2001*, Korea, May 2001, pp. 624–629.

[5] K. A. De Jong and W. M. Spears, "Using genetic algorithms to solve NP-complete problems," in *Proc. 3rd Int. Conf. Genetic Algorithms*, Arlington, VA, Jun. 1989, pp. 124–132.

[6] B. Rylander, T. Soule, J. Foster, and J. Alves-Foss, "Quantum evolutionary programming," in *Proc. Genetic Evol. Comput. Conf. (GECCO-2001)*, San Francisco, CA, Jul. 2001, pp. 1005–1011.

[7] K. H. Han and J. H. Kim, "Quantum-inspired evolutionary algorithm for a class of combinatorial optimization," *IEEE Trans. Evol. Comput.*, vol. 6, no. 6, pp. 580–593, Dec. 2002.

[8] M. Lukac and M. Perkowski, "Evolving quantum circuits using genetic algorithm," in *Proc. 2002 NASA/DoD Conf. Evolvable Hardware*, Alexandria, VA, Jul. 2002, pp. 177–185.

[9] H. De Garis, A. Gaur, and R. Sriram, "Quantum versus evolutionary systems: Total versus sampled search," in *Proc. 5th. Int. Conf. Evolvable Syst. (ICES)*, Trondheim, Norway, Mar. 2003, pp. 457–465.

[10] G. A. Giraldi, R. Portugal, and R. N. Thess, *Genetic Algorithms Quantum Comput.* 2004. [Online]. Available: http://www.arxiv.org/pdf/cs.NE/0403003

[11] G. Zhang, L. Hu, and W. Jin, "Resemblance coefficient and a quantum genetic algorithm for feature selection," *Lecture Notes Comput. Sci.*, vol. 3245, pp. 155–168, Jan. 2004.

[12] M. Martinez, L. Longpre, V. Kreinovich, S. A. Starks, and H. T. Nguyen, "Fast quantum algorithms for handling probabilistic, interval, and fuzzy uncertainty," in *Proc. 22nd Int. Conf. North Amer. Fuzzy Inf. Process. Society, NAFIPS 2003*, Chicago, IL, Jul. 2003, pp. 395–400.

[13] T. C. Weinacht and P. H. Bucksbaum, "Using feedback for coherent control of quantum systems," *J. Opt. B*, vol. 4, no. 3, pp. R35–R52, 2002.

[14] G. Turinici, C. Le Bris, and H. Rabitz, "Efficient algorithms for the laboratory discovery of optimal quantum controls," *Phys. Review E*, vol. 40, no. 1, p. 016704, 2004.

[15] V. Petridis, S. Kazarlis, and A. Bakirtzis, "Varying fitness functions in genetic algorithm constrained optimization: The cutting stock and unit commitment problems," *IEEE Trans. Syst. Man Cybern. B*, vol. 28, no. 5, pp. 629–640, Oct. 1998.

[16] C. Dürr and P. Høyer, *A Quantum Algorithm For Finding Minimum*, 1996. [Online]. Available: http://www.arxiv.org/pdf/quantph/9607014

[17] J. H. Holland, *Adpatation in Natural and Artificial Systems*. Ann Arbor, MI: Univ. of Michigan Press, 1975.

[18] H. Mühlenbein and D. Schlierkamp-Voosen, "Predictive models for the breeder genetic algorithm," *Evol. Comput.*, vol. 1, no. 1, pp. 25–49, 1993.

[19] H. Mühlenbein and D. Schlierkamp-Voosen, "The science of breeding and its application to the breeder genetic algorithm," *Evol. Comput.*, vol. 1, no. 1, pp. 335–360, 1994.

[20] A. Peres, *Quantum Theory: Concepts and Methods*. Norwell, MA: Kluwer, 1998.

[21] C. H. Bennett, G. Brassard, C. Crepeau, R. Jozsa, A. Peres, and W. Wootters, "Teleporting an unknown quantum state via dual classical and EPR channels," *Phys. Rev. Lett.*, vol. 70, no. 13, pp. 1895–1899, Mar. 1993.

[22] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge, U.K.: Cambridge Univ. Press, 2000.

[23] M. Boyer, G. Brassard, P. Høyer, and A. Tapp, "Tight bounds on quantum searching," *Fortschr. Phys.*, vol. 4, no. 5, pp. 493–505, 1998.

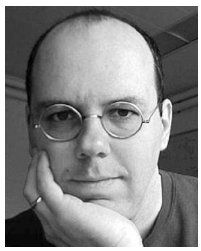
[24] A. Malossini, "Un algoritmo genetico di ricerca quantistica," M.S. thesis, University of Padova, Padova, Italy, Jul. 2002.

Andrea Malossini received the Laurea degree (M.S.) in physics from the University of Padua, Padua, Italy, in 2002, and the Ph.D. degree in computer science from the University of Trento, Povo, Trento , Italy, in 2007.

His main research interests focus on bioinformatics, in particular, on the analysis of biological and medical data sets by means of statistical learning techniques. As a consequence of his physical background, he maintains a keen interest in the field of quantum computation.

Enrico Blanzieri was born in Ferrara, Italy, in 1965. He received the Laurea degree (*cum laude*) in electronic engineering from the University of Bologna, Bologna, Italy, in 1992, and the Ph.D. in cognitive science from the University of Turin, Turin, Italy, in 1998.

From 1997 to 2000, he was a Researcher at ITC-IRST of Trento, and from 2000 to 2002, he was an Assistant Professor with the Faculty of Psychology, University of Turin. Since 2002, has been an Assistant Professor with the Department of Information and Telecommunication Technology, University of Trento. His research interests include soft computing, artificial intelligence, and bioinformatics.



Tommaso Calarco is a Senior Researcher at the CNR-INFM Center for Bose-Einstein Condensation, Trento, Italy. After his Ph.D. degree in hadron physics in 1998, his main research interests focus on the physical implementation of quantum information processing with atomic systems. He worked in some of the major research centers in this field—as a Fulbright Fellow at the U.S. National Institute of Standards and Technology, and as a Marie-Curie Fellow both at Innsbruck University and at Harvard University.